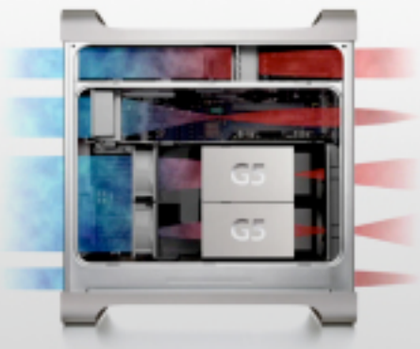
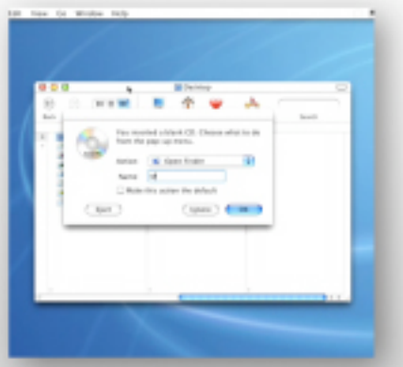
Bluetooth Update

Session 501





Bluetooth Update

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Bluetooth Update

Eric Brown
Bluetooth Software Engineer
esb@apple.com

Last Year

Jaguar—Mac OS X 10.2 == Apple Bluetooth 1.0

- Profiles
  - Serial Port Profile
  - Dial-up Networking
  - Object Push
- Applications
  - Bluetooth File Exchange
  - System Preferences
  - Menu Extra

Last Year

- APIs
  - IOBluetooth.framework
    - IOBluetoothDevice
    - IOBluetoothL2CAPChannel
    - IOBluetoothRFCOMMChannel
  - OBEX
  - SDP

Last Year

- APIs
  - IOBluetoothUI.framework
    - IOBluetoothDeviceSelectorController
    - IOBluetoothServiceBrowserController
    - IOBluetoothPairingController

What Is New

- Profiles
  - File Transfer Profile (FTP)
  - HID Profile
- Applications
  - Bluetooth Setup Assistant
  - Bluetooth Serial Utility
  - Reworked preferences pane

What Is New

- APIs
  - New/updated APIs on the following objects
    - IOBluetoothDevice
    - IOBluetoothL2CAPChannel
    - IOBluetoothRFCOMMChannel
  - UI elements
- Continued parity between ObjC and C APIs

HID Support

- Apple provides a complete HID class driver
- HID Manager is the proper place to get access to the data
- No direct access to the Bluetooth channels (L2CAP) from Kernel
- Contact: Craig Keithley—keithley@apple.com

Bluetooth Serial Ports

- Bluetooth Serial Utility
  - Settings are a little complex for the novice user
  - Only way to create Bluetooth serial ports
- Two types of serial ports
  - Incoming
  - Outgoing

Incoming Serial Ports

- Any device can connect to the serial port service
- Can be opened at any time without a connection to a remote device
  - Reads return no data, writes get dropped on the floor
  - Reads and writes get connected to actual data when a remote device connects (Palm Hot-sync)
  - Behaves like a serial port with no cable attached

Outgoing Serial Ports

- User can configure a serial port to any RFCOMM service on a remote device
- System attempts to connect to a device and open the RFCOMM channel when the port is opened
- Failure to open a connection or channel results in a failure to open the serial port
- Opening outgoing port can be a lengthy synchronous process for the client application

Direct Serial Access

- Serial ports can be accessed directly by using RFCOMM APIs
  - Better error reporting
  - More granular error codes
  - Direct control over the creation and behavior of the RFCOMM channel
  - More information available about the channel
  - Recommended

UI Element Changes

- New device categories
  - Favorites
  - Recent devices
- UI changes
  - Filter by
    - Device type (computers, phones, PDAs, etc...)
    - Device category (favorites, recent or discovered devices)

UI Element Changes

- New API consistent across all UI objects
- New developer functionality
  - Run panel modal or as a sheet
  - Title, prompt and description customizable
  - Filter and validate user selection by UUID
- Headerdoc for all of the new UI element API

Device Attributes

- On IOBluetoothDevice object:
  - Favorite device attributes
    - + (NSArray *)favoriteDevices
    - - (BOOL)isFavorite
    - - (IOReturn)addToFavorites
    - - (IOReturn)removeFromFavorites
  - Paired device attributes
    - + (NSArray *)pairedDevices
    - - (BOOL)isPaired

Device Attributes

- On IOBluetoothDevice object:
  - Recent device attributes
    - Find devices by the time/date of the last access
    - + (NSArray *)recentDevices:(UInt32)numDevices
    - - (NSDate *)recentAccessDate

New Async APIs for L2CAP/RFCOMM

- New fully async APIs
  - Open L2CAP and RFCOMM channels
  - Async write APIs on L2CAP and RFCOMM channels
  - Notification of changes in channel state
- New APIs are available from 10.2.5 onward
- Highly recommended
- Improve application responsiveness
- Avoid spinning cursor

Async L2CAP APIs

- New L2CAP channel delegate to receive async notifications and events:
  - New incoming data
  - Channel open complete
  - Channel closed
  - Channel reconfigured
  - Write complete

Async L2CAP APIs

- IOBluetoothDevice methods
  - -openL2CAPChannelAsync:withPSM:delegate:
  - -openL2CAPChannelSync:withPSM:delegate:
  - Delegate must be specified when opening new L2CAP channels
  - Old method: -openL2CAPChannel:findExisting:newChannel: to be deprecated

Async L2CAP APIs

- IOBluetoothL2CAPChannel methods:
  - -writeAsync:length:refCon:
    - Delegate called when async write completes
  - -writeSync:length:
  - Replaces old -write:length:

Async RFCOMM APIs

- New RFCOMM APIs similar to L2CAP APIs with delegate to receive async notifications and events:
  - New incoming data
  - Channel open complete
  - Channel closed
  - Control signals and flow control changed
  - Write complete
  - Queue space available

Async RFCOMM APIs

- IOBluetoothDevice methods:
  - -openRFCOMMChannelAsync:withChannelID:delegate:
  - -openRFCOMMChannelSync:withChannelID:delegate:
  - Delegate must be specified when opening a new RFCOMM channel
  - Old method: -openRFCOMMChannel:channel: to be deprecated

Async RFCOMM APIs

- IOBluetoothRFCOMMChannel APIs
  - -writeAsync:length:refCon:
  - -writeSync:length:
  - Deprecated methods:
    - -write:length:sleep:
    - -writeSimple:length:sleep:
  - -registerIncomingDataListener:refCon:
  - -registerIncomingEventListener:refCon:

Async RFCOMM APIs

- IOBluetoothRFCOMMChannel queue management
  - RFCOMM flow control dictates when we are allowed to send data
  - Internally we will queue data when sending is disallowed
  - Queue space is limited so it is possible a write call will fail with a queue full error (kIOReturnNoSpace)
  - When queue space becomes available, the -rfcommChannelQueueSpaceAvailable: delegate message is sent

Local SDP Services

- Two types of services
  - Transient
    - Service is removed when the client that registered it exits
  - Persistent
    - Service exists until explicitly removed
    - Survives reboots
    - Does not require a client application to be running
    - May launch client application when a new connection to the service is requested

SDP Service Description

- Each SDP service is described by a dictionary
  - Each entry in the dictionary represents an SDP attribute
    - Dictionary and attribute format described in headerdoc
  - Some attributes may be assigned by the system
    - Service record handle
    - RFCOMM channel ID
  - Additional “LocalAttributes” entry may be present to control the behavior of the service
    - “LocalAttributes” entry is a dictionary of local attributes

Local Attributes

- Currently only two supported attributes
  - **Persistent**
    - Boolean attribute must be TRUE for the service to persist beyond the life of the client that added it
  - **TargetApplication**
    - Full path to the target executable to launch when an incoming connection to the service is received
  - More useful for persistent services
- More attributes may be added in the future

Adding a New SDP Service

- IOBluetoothAddServiceDict()
  - Dictionary format makes it easy to define service in a plist file
    - Use -[NSDictionary dictionaryWithContentsOfFile:]
  - Returns a new IOBluetoothSDPServiceRecord object which includes system-assigned record handle and channel IDs
  - Client must save the record handle to be able to explicitly remove the service later

Removing an SDP Service

- IOBluetoothRemoveServiceWithRecordHandle()
  - Must use record handle returned in original service
  - Transient services will be removed automatically when client exits
  - Transient services may call this function to remove service before client exit

Helpful Hints

- Utilize example code
  - RFCOMMServerSample
- Possible to remove services accidentally left on development machine
  - Remove ~root/Library/Preferences/blued.plist
  - Warning—will also remove device cache information
    - Device names
    - Device services
    - Paired device information

SDK

- Latest released APIs from Apple
- PacketDecoder is disabled by default in production
- Security is compromised
  - Link Keys
  - HID data
- SDK includes a build of the stack which supports PacketDecoder

Bluetooth Industry Directions

Michael Larson
Bluetooth Technology Manager
larson@apple.com

Bluetooth Industry Notes

- Large push for simpler configuration, better interoperability
  - SIG—5 minute ready initiative. Consumer should be able to use the device inside of 5 minutes.
  - Others believe it needs to be shorter
- User Interface
  - Bluetooth is coming to the foreground instead of being buried on devices. SonyEricsson T610 is much better and easier to use than any phone before it.

Bluetooth SIG Upcoming Releases

- Late 2003—Version 1.2
  - Improved QualityOfService (QoS) implementation
  - Implementation of Adaptive Frequency Hopping (AFH)
    - Improves co-existance with other 2.4GHz devices
    - Version 1.2 radio required on both sides
  - Many radios are firmware upgradable to 1.2 versions of spec
    - You want a BT 1.2 compatible radio at this time

Bluetooth SIG Upcoming Releases

- Medium Data Rate
  - Backwards compatible solution to deliver between 2–3 Mbit throughput to other compatible radios
  - Currently scheduled for mid-2004
  - Will require HW changes from today's radios

Bluetooth Best Practices

- Inquiry and paging operations really disrupt 802.11 traffic
- Direct inquiry APIs are not planned
- Bandwidth consumption
  - 720 kbps is the total budget
  - It is shared between everything (2xHID, Phone Sync, Dial-up)
    - Be considerate of this before choosing Bluetooth as the transport for your product

More Bluetooth Best Practices

- Avoid a constant loop of attempting to connect to a device:
  - Not a responsible use of resources
  - Severe performance loss on 802.11 side
  - Interrupts standard Bluetooth performance as well

Bluetooth Opportunities

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Opportunities

- Low-cost Bluetooth serial adapter
  - Needs to be under $75
- HID devices
  - Keyboards, mice, trackpads, gamepads
- Digital still cameras
  - PTP over Bluetooth
- GPS receivers

Bluetooth HW Solutions

- HW solutions often include an embedded Bluetooth stack to handle your profile needs:
  - Examples: Serial, Advanced Audio, etc.
  - If you have specialized needs for a device contact:
    - Craig Keithley—keithley@apple.com

Roadmap

500 Wireless Directions

Nob Hill
Mon., 3:30pm

Who to Contact

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Michael Larson
Bluetooth Technology Manager
larson@apple.com

Eric Brown
Bluetooth Software Engineer
esb@apple.com

Bluetooth Developer Mailing List
Discuss Bluetooth with other developers
http://www.lists.apple.com/mailman/listinfo/bluetooth-dev

http://developer.apple.com/wwdc2003/urls.html

Reference Library

Documentation

Bluetooth
http://developer.apple.com/hardware/bluetooth

Bluetooth Membership Site
http://www.bluetooth.org

Official Bluetooth Website
http://www.bluetooth.com

Q&A

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference